Simulation Analysis of the Optimal Storage Resource Allocation for Large HENP Databases

Jinbaek Kim and Arie Shoshani
Lawrence Berkeley National Laboratory
(jinbaek@ieor.berkeley.edu, shoshani@lbl.gov)

Abstract

Large High Energy and Nuclear Physics (HENP) databases are commonly stored on robotic tape systems because of cost considerations. Later, selected subsets of the data are cached into disk caches for analysis or data mining. Because of the relatively long time to mount, seek, and read a tape, it is important to minimize the number of times that data is cached into disk. Having too little disk cache will force files to be removed from disk prematurely, thus reducing the potential of their sharing with other users. Similarly, having too few tape drives will not make good use of a large disk cache, as the throughput from the tape system will form the bottleneck. Balancing the tape and disk resources is dependent on the patterns of the requests to the data. In this paper, we describe a simulation that characterizes such a system in terms of the resources and the request patterns. We learn from the simulation which parameters affect the performance of the system the most. We also observe from the simulation that, there is a point beyond which it is not worth investing in additional resources as the benefit is too marginal. We call this point the "point-of-no-benefit" (or PNB), and show that using this concept we can more easily discover the relationship of various parameters to the performance of the system.

1. Introduction

HENP experiments produce large datasets by simulation or collect large amount of data from detectors. The size of such datasets can reach 100's of terabytes a year, and therefore they are not stored on disk systems, but rather on robotic tape systems. After reconstruction, datasets of similar sizes are usually generated as well. Typically, each dataset is partitioned into files that are large enough for the efficient use of the robotic tape system (100's of MBs per file). Selected files are then downloaded into disk caches upon request for analysis. Often, only parts of these files are needed, but entire files are downloaded. We consider here the case that multiple users (processes) share a single large disk cache. In such a case, the cache may be full when a new request to download files arrives, and the process has to wait till space on the disk cache can be freed. Similarly, the process may have to wait if the tape drives on the robotic tape system are busy at the time the request is made. If the system is overloaded, increasing the number of tape drives will improve the performance of the system and so will increasing the disk cache size. However, the coordination of the two resources is necessary, since there may be little benefit in increasing each type of resources beyond a certain level. Given a limited budget, a very practical question is what resources to spend the budget on, more tape drives or more disk cache.

The analysis of this problem is hard because it depends on the access patterns of the file caching requests. We call a request for multiple files a "query". Many parameters and policies are involved in modeling such a system, and therefore, it is hard to formulate a precise analytical solution. In this paper, we describe the results of a series of simulations that we performed in order to understand the qualitative and quantitative dependency between the various parameters and the optimal resource allocation.

Our simulation model was motivated by typical HENP applications (e.g. [1]). The typical aspect of the analysis phase is that when data is accessed, event objects can be analyzed in any order, and several techniques to support such access have been developed [2,3]. There were various aspects of tertiary storage system studied in the literature, such as modeling their performance characteristics (e.g. [4]), space organization (e.g. [5]), and scheduling tape access (e.g. [6]). Our interest is different: determining how many drives, and how much disk cache to use given a request profile.

2. Model description and parameter definitions

Figure 1 shows a schematic diagram of the system being modeled. We assume that queries arrive with a specified inter-arrival time. A file requested by a query may be currently stored on tape or reside in the disk cache if it was previously read by another query. If a file requested by a query is already in cache, the processor assigned to process the query that requested this file can start processing the file right away. If the file is not in the cache, a tape drive loads it into cache as soon as possible, and then the processor can start processing the file. Only one processor is assigned to each query. Queries can have files in common. This is typical for users that are analyzing similar regions in the dataset. When a query is brought into cache, and 2 or more queries can use the same file simultaneously. We also assume in this model that no parallel processing is performed by queries, i.e. queries that are busy processing a file cannot start processing another file until they are finished with the current file.

2.1 Parameter definitions

In this section we define the various parameters that are used in the model to characterize the resources and query patterns.

- Read Speed [R]: the average time to read a file from tape and to write it into cache.
- Processing Time [T_p]: the time to process a file that is already in the cache.
- Number of Queries [Q]: the number of queries used in a simulation.
- Number of Files in a Query [F]: the average number of files needed by a query.

- File Size [S_f]: the average file size
- Cache Size [S_c]: the size of the cache in GBs.
- Number of Drives [*D*]: the number of tape drives in the tape robot.
- Overlap Factor [F_o]: the fraction of the number of files that overlap between any two queries. (We will explain below how files are assigned to queries to achieve this.)
- Clustering Factor [F_c]: percent of data read from a file on the average by each query.
- Processing Rate [ρ]: the number of seconds required to process 1 MB.
- Processing time [T_p]: the time to process a file on the average, in seconds.

Note: The processing time is determined by using the clustering factor (F_c), the file size (S_f), and the processing rate (ρ). The following formula holds: $T_p = \rho \times S_f \times F_c$. For example, if the processing rate is 2 seconds/MB, the file size is 1 GB, and the clustering factor 1% (or 0.01) we get that the processing time T_p for a 1 GB file = 2 *1000 * 0.01 = 20 seconds.

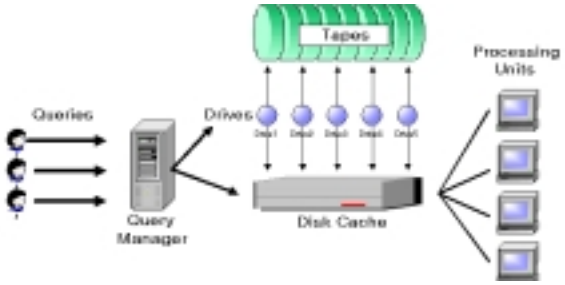

Figure 1

2.2 Discussion of parameter choices

We had to make some choices on which parameters to fix in the simulations and which to vary in order to make the number of simulation runs reasonable. We decided to fix some of the parameters at reasonable values, based on the application information. We ended up choosing a fixed average value for reading a file from tape, fixing the file size, as well as the query inter-arrival time. The parameters we varied are: 1) the number of drives, 2) the disk cache size, 3) the processing time per file, 4) the overlap factor, 5) the number of queries. To analyze the variations of these 5 parameters, we ended up with about 4000 runs on MATLAB to reach the conclusions reported here. The parameter choices for each run will be explained in section 4.

We assume the following figures to calculate the read speed: 20 seconds for mounting a tape, 30 seconds for seeking on the average to the middle of a 30 GB tape at 500 MB/sec, and a read rate of 5 MB/s (200 seconds for a 1 GB file). Thus, for a 1 GB file, the total read time is 250 seconds on the average.

2.3 Simulating the overlap factor

We wanted the overlap factor between any two queries to be the same. Given that the number of files per query is given, and the desired overlap factor we want to use for a simulation, we had to start with the certain number of files to choose from at random. This was achieved as follows. Let the number of files in a query be F, and the total number of files to choose from N, where F < N. If we choose files at random to be included in a query, then the probability that a particular file is selected is F / N . The probability that a particular file is selected again by another query is $(F / N)^2$. The overlap factor, F_o , is the probability that any two queries have the same file, and therefore $F_o = (F / N)^2$ or $N = F / \sqrt{F_o}$. For example, for a set of queries that have 100 files each, and a 1% Overlap Factor (F_o) we start with 1000 files ($100 / \sqrt{0.01} = 1000$), and randomly select 100 files out of 1000 to populate each query.

2.4 Policies

Our focus in this simulation is the performance of the system rather than seeing the effect of various policies. For this simulation, we chose to model a system that is "fair" to the users, so we selected a simple round-robin policy. Queries are assigned a number according to their arrival order. Each query is given one file at a time, and can request another file only after they process the current file. Files can be shared by queries and processed by queries in any order. Therefore, when it is some query's turn to be serviced, and there is a file for that query in cache, that file is given to that query. Otherwise, a drive is assigned to the query in order to read a file for it into cache. If we make one round over the queue and find that all the queries are busy processing, we stop till one of the queries is finished processing, and proceed to service it.

There are three considerations that determine the caching policy:

1. Which file to read first into cache, when more than one file is requested by the query. Our choice: to stage files in the order requested, unless a file is found in cache for that query.
2. Which file (not currently in use) to remove from cache, when the cache is full. Our choice: from the set of files that are not currently in use, the file that has been in the cache the longest time (the "oldest" file) is removed first. This is equivalent to "last in first out" (LIFO) policy.
3. Which file to assign to a query, when more than one file for that query is in the cache. Our choice: select the file that has been longest in the cache, so as to prevent it from being removed.

3. The performance measure

A typical performance measure is the average time to execute a collection of queries. This is an absolute measure. However, we chose to consider a relative measure (i.e. a normalized measure) where the performance of the system is measured relative to the "ideal" performance. The ideal time to perform a query is obtained when we assume that all the files of each query are already in disk cache. This is a measure that represents the best time expected to execute a query if it was the only query on the system. In this case, except for the first file that is brought from tape, all the other files can be assumed to be in disk. This is because while the first file is being processed, there are enough resources in this "ideal" system to bring the additional files into cache, etc.

The usefulness of a measure relative to the ideal time is that it is more meaningful to a user. For example, an absolute execution time improvement of 500 sec is a lot more significant when the "ideal" execution time is 1,000 sec than if the ideal time was 10,000 seconds. The relative measure is called "efficiency".

Definition: Efficiency

The efficiency of a query *i* is defined as the ratio between the time that it actually took to execute a query (T_{actual}) and the time that would have taken if all its files were already loaded into disk cache (T_{ideal}). T_{ideal} is the product of the number of files of the query (*F*) and the processing time per file (T_p). Thus, the efficiency of query *i* is: $efficiency(i) = \frac{T_{actual}(i)}{T_{ideal}} = \frac{T_{actual}(i)}{F \times T_p}$. The efficiency of the system is defined as the average over all the queries running on the system over a certain period of time: $efficiency = \frac{\sum_{i=1}^{Q} efficiency(i)}{Q}$

3.1 The points-of-no-benefit (PNB)

For each simulation, we chose a number of tape drives, and varied the cache size in small increments (10 MB increments). This was then repeated for the next increment in the number of tape drives. An example is shown in Table 1. We observed from the simulations that for a given number of tape drives, it was not beneficial to increase the size of the disk cache beyond a certain point as the increase in the efficiency of the system was marginal. We called this point the "point-of-no-benefit" (PNB). Each PNB has the pair of values for the number of drives and the size of disk cache (*D*, S_c) associated with it.

Using this concept we could now analyze the simulation results of these large number of runs, by focusing only on PNB points to interpret the results. Intuitively, a PNB represents the behavior of the system at a "balanced state", in that the amount of cache used is balanced for the number of drives.

Every simulation step involves some combination of the storage system resources (number of drives, amount of disk) and the query patterns (overlap factor, processing time, etc.). The simulation results in some efficiency figure. In Table 1, we show the results of many such simulations, where the columns represent an increasing number of tape drives, and the rows represent increasing cache size values. The table was produced for a certain query patterns, where the overlap factor is set to 0.1, the processing time is 100 sec per file, and the number of queries Q is 20.

The PNBs can be clearly seen in Table 1, when we go down each column. For example, if the system has only one drive, no improvement in efficiency is gained beyond 80 GBs of cache. Similarly, for 2 drives, no improvement is gained beyond 150 GBs, etc. We note that the PNB depends on the precision of the efficiency measure. For example, in Table 1, the precision is the 4th decimal place, i.e. 0.0001. We can consider the precision as a *threshold* beyond which no improvement is achieved. For each Query profile, there are many such points. For example in Table 1, we have 5 pairs: (1 drive, 80 gigabytes), (2 drives, 150 gigabytes), (3 drives, 230 gigabytes), (4 drives, 300 gigabytes), (5 drives, 310 gigabytes), etc. Intuitively, this behavior of PNBs is to be expected, since as we increase the disk cache, eventually it does not help since the number of drives is the limiting factor. However, looking at the series of PNBs, this reveals the points where even increasing the number of drives provides only a marginal improvement of the efficiency for a particular query profile. We will use the PNB points to plot the behavior of the system in section 5. More formally, we have the following definition.

	Number of Drives, *D*				
S_C (GB)	*D=1*	*D=2*	*D=3*	*D=4*	*D=5*
10	8.4725	4.6087	3.3325	2.5982	2.0600
20	8.2350	4.4612	3.2564	2.5355	1.9831
30	8.0325	4.3887	3.1672	2.4807	1.9127
40	7.7425	4.2938	3.0643	2.3988	1.8259
50	7.5150	4.1850	2.9587	2.3079	1.7413
60	7.2200	4.0481	2.8513	2.1701	1.6662
70	7.0425	3.9588	2.7842	2.0669	1.5672
80	**6.9575**	3.7881	2.6512	2.0187	1.4779
90	6.9575	3.6562	2.5460	1.9028	1.4179
100	6.9575	3.5106	2.4460	1.8135	1.3625
110	6.9575	3.3789	2.3449	1.7427	1.3222
120	6.9575	3.3025	2.2575	1.6263	1.2863
130	6.9575	3.2431	2.1972	1.5678	1.2674
140	6.9575	3.1400	2.1084	1.5166	1.2443
150	6.9575	**3.0800**	2.0509	1.4615	1.2368
160	6.9575	3.0800	1.9507	1.4421	1.2299
170	6.9575	3.0800	1.9052	1.4258	1.2214
180	6.9575	3.0800	1.8829	1.4141	1.2174
190	6.9575	3.0800	1.8521	1.4103	1.2142
200	6.9575	3.0800	1.8291	1.4063	1.2080
210	6.9575	3.0800	1.8178	1.4009	1.2072
220	6.9575	3.0800	1.8157	1.3968	1.2025
230	6.9575	3.0800	**1.8145**	1.3944	1.1998
240	6.9575	3.0800	1.8145	1.3907	1.1975
250	6.9575	3.0800	1.8145	1.3871	1.1943
260	6.9575	3.0800	1.8145	1.3852	1.1896
270	6.9575	3.0800	1.8145	1.3842	1.1878
280	6.9575	3.0800	1.8145	1.3829	1.1861
290	6.9575	3.0800	1.8145	1.3810	1.1829
300	6.9575	3.0800	1.8145	**1.3807**	1.1816
310	6.9575	3.0800	1.8145	1.3807	**1.1796**
320	6.9575	3.0800	1.8145	1.3807	1.1796

Table 1. Efficiency (F_o =0.1 and T_p =100, Q=20)

Definition: The Point of No Benefit (PNB)

Given a "threshold" chosen for the simulation, and the number of drives *D*, the PNB(*D*) is defined as the pair (*D*, S_c), such that: (efficiency (*D*, S_c) - efficiency (*D*, S_c +1)) < threshold,

where S_c +1 is the next increments of file size after S_c .

The efficiency at this point is called the "PNB-efficiency". In the next sections, we only use the PNB points and the corresponding PNB-efficiency at these points to analyze the results of the simulations.

4. The Simulation Setting

Our approach for the simulation setting is to select a query pattern (in terms of processing time per file T_p , file overlap F_o , the number of files per query *F*, and the number of queries per run *Q*. For each such pattern, we ran a series of simulations for various values of the number of drives *D* and the size of cache S_c . Each simulation generated the efficiency. This produces a single table, similar to Table 1. We then proceed to choose other values for the query pattern to produced additional tables. The parameters used were:

[T_p = 10 - 1000 seconds per file]. We used 10, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000.

[*F* = 100-300 files per query]. We used 100, 200, 300.

[*Q* = 20-60 queries per run]. We used 20, 40, 60.

[F_o = 0.00001 - 0.1]. From practically no overlap (0.00001) to a high level of overlap (0.1).

[D = 1-5]. We observed during the simulations that the dependency behavior was sufficiently clear by simulating up to 5 drives. Therefore, we did not need to use a larger number of drives.

5. Qualitative Analysis

Our purpose in this section is to discuss our conclusions about the qualitative behavior of the system. We first examine the dependency of cache size as a function of the number of drives used at PNBs. This is followed by an analysis of the effect of various parameters on the efficiency of the system.

5.1 Cache size as a function of the number of drives at PNBs.

We examine two cases in Figure 2: a high file overlap and a low file overlap. The reason for choosing these cases is that at first glance one would expect that a high file overlap would require less disk cache. However, the simulation showed the opposite result. This is because a high file overlap requires files to stay in cache longer for the files to be shared, and therefore we need more cache. We also observe that the need for more cache was very sensitive to the processing time. Only for small processing times, we could benefit from more cache. This is shown in Figure 2 below. We observe that: (i) when the overlap factor is large, the general trend is that the required cache size grows with the number of drives. However, this is especially true for queries with short processing time; (ii) when the overlap factor is small, the cache size requirements are generally lower. That can be explained by observing that we cannot benefit from having more cache since files are not shared in the case of little or no overlap.

In summary, we can see that the cache size at PNB points generally increases or stays flat as D grows larger. The cache size increases faster when processing time is short, and when overlap is high.

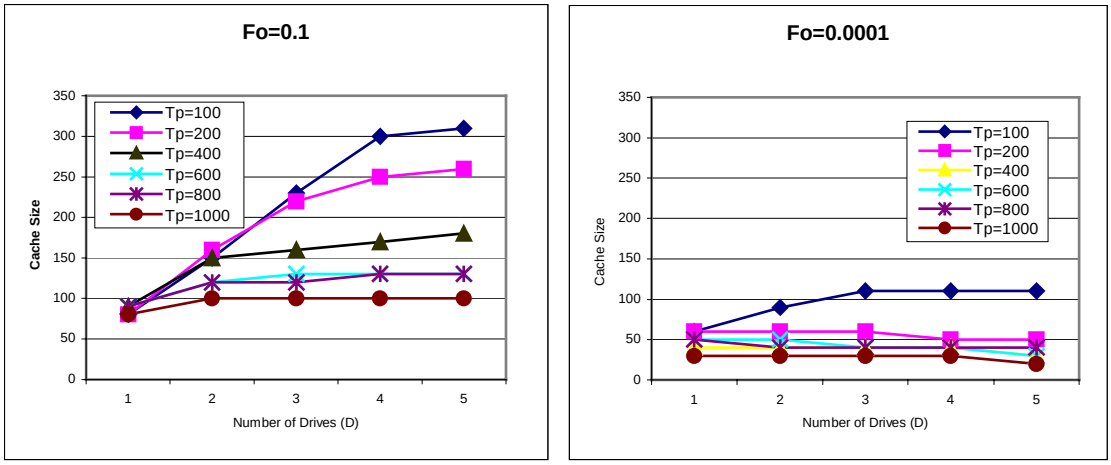

Figure 2: The relationship between number of drives and cache size at PNBs

5.2 The effect of various parameters on efficiency

In this section we want to see the effect of increasing resources on the efficiency. Specifically, we plot the efficiency vs. the amount of resources (represented by a combined $ cost of drives and disk (C), and vary the overlap factor (F_o), the processing time (T_p), and the number of queries (Q), and the file size (F).

We speculated that the numbers of files per query, F, will have little or no effect on the efficiency because of the following reason. The ideal time to execute a query increases proportionally with the number of files in the query. But the actual execution time also increases proportionally with the number of files in a query. Thus the efficiency (which is a relative measure) changes very little. This was verified by the simulation results regardless of values of the other parameters. We do not show these graphs here as they are not interesting and because of space limitations.

Since it is hard to visualize the effect of all the 4 remaining parameters at once, we chose to plot the efficiency as a function of the amount of resources (measure in $ cost) having multiple lines for varying values of processing time (T_p). This is shown in Figure 3. To see the effect of the other 2 parameters we plot the same graphs for 2 extreme values of the overlap factor (F_o) and the number of queries (Q).

The resources at each PNB point include the number of tape drives, D, and the amount of disk cache, S_c. In order to use a single figure for the resources, we chose some reasonable cost for the drives and a unit of disk, and computed a joint cost figure using: $C_T = c_d \times D + c_c \times S_c$ where c_d is the cost of a tape drive and c_c is the cost of a unit of disk. We used c_d =\$10,000 and c_c =\$5/GB in Figure 3 below. (These cost figures are a bit optimistic; indications are that these costs will be realistic in a year or two.)

In Figure 3, we present four plots. Each plot has a series of lines that show the efficiency against the amount of resources at different processing times. By comparing two plots from the left to the right, we can see the effect of the number of queries. Similarly, by comparing the two plots from the top to the bottom, we can see the effect of the overlap factor.

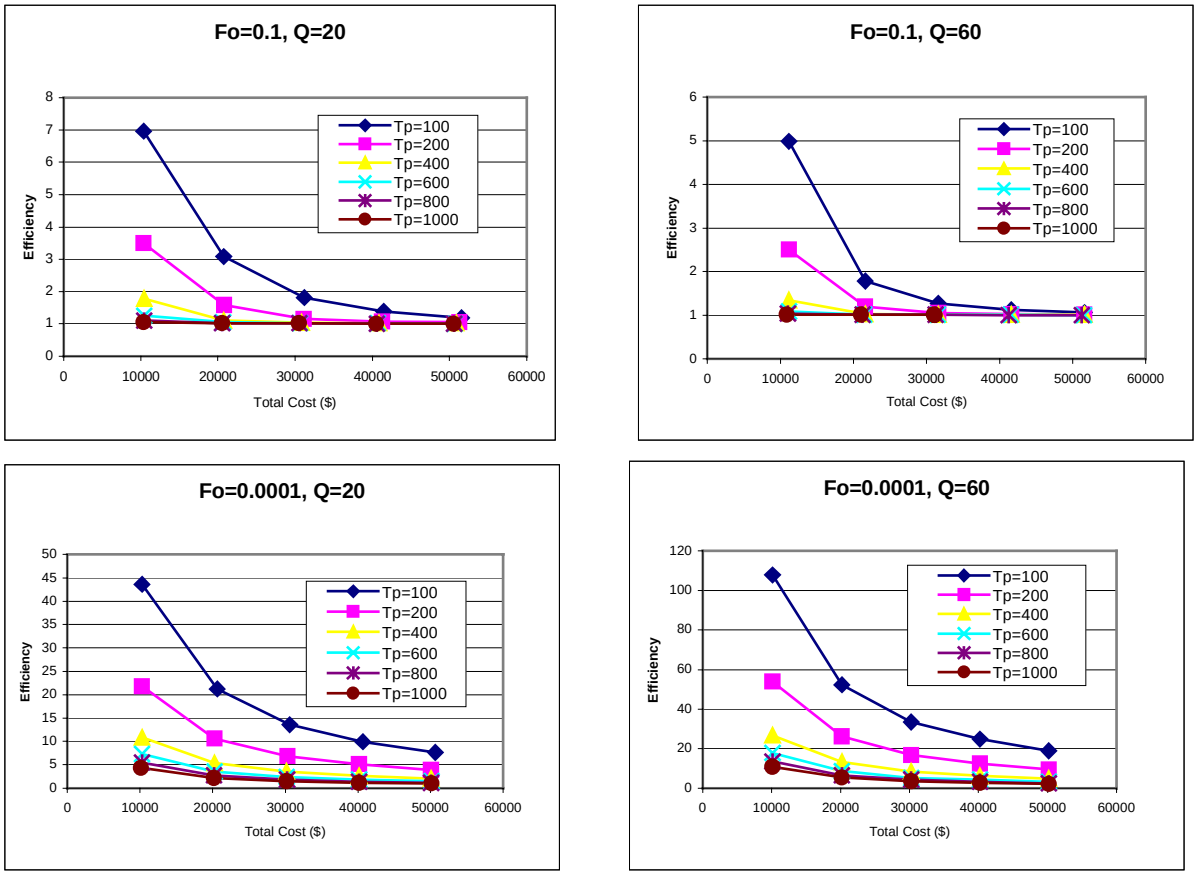

Figure 3: efficiency as a function of resources ($ cost)

As can be seen from Figure 3, the efficiency generally improves (i.e. its value decreases) with the increase in resources, as expected. However, as we increase the resources the benefit in efficiency diminishes. We can see that the benefit is most pronounced when the processing time is short. For large processing times, there is little benefit in going beyond D=1. This behavior is intuitively expected, since if processing time is large, there is plenty of time for bringing files to cache, and very few resources are sufficient.

We also note that the efficiency is largely affected by the overlap factor. Comparing the two leftmost plots, we observe that the efficiency values are significantly larger (i.e. performance is worse) for small overlap. The efficiency deteriorates almost linearly (i.e. it is 3 times larger). In summary, when file overlap is very small the relative benefit of increasing resources is even larger, especially with a large number of queries. In summary, the benefit of adding resources is most sensitive to query profiles with short processing time and low overlap factor.

Our methodology based on PNBs provides a simple technique can be used to design the system. Given the query profiles, we run the simulation to determine the PNB points, each for an additional tape and the corresponding balanced disk size. When the incremental cost of going from one PNB to the next is too high for the efficiency benefit achieved, we stop. This lat PNB represents the desired level of tape and disk resource for that query profile.

References

[1] The STAR Collaboration, http://www.rhic.bnl.gov/STAR/.

[2] Luis M. Bernardo, Arie Shoshani, Alex Sim, Henrik Nordberg: Access Coordination of Tertiary Storage for High Energy Physics Applications. IEEE Symposium on Mass Storage Systems 2000: 105-118.

[3] Koen Holtman, Peter van der Stok, Ian Willers: A Cache Filtering Optimisation for Queries to Massive Datasets on Tertiary Storage. DOLAP 1999: 94-100.

[4] Bruce Hillyer, Abraham Silberschatz: On the Modeling and Performance Characteristics of a Serpentine Tape Drive. SIGMETRICS 1996: 170-179.

[5] Ling Tony Chen, R. Drach, M. Keating, S. Louis, Doron Rotem, Arie Shoshani: Efficient organization and access of multi-dimensional datasets on tertiary storage systems. Information Systems 20(2):155-183 (1995).

[6] Sachin More, Alok N. Choudhary: Scheduling Queries for Tape-Resident Data. Euro-Par 2000: 1292-1301.